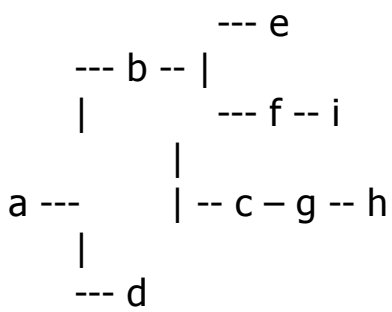
# System programming (03KTOOT)

## Laboratory #1: **UNIX shell**

Laboratories are mandatory. Each student must submit the work (exclusively in .zip format, with text in .pdf and code (if required) in a format able to be run on the LABINF computers) following the rules published on the web portal. Please remember that for each laboratory the final mark can get from -2.0 (not submitted or with severe flaws) up to +0.5 points (excellent work). If all 4 laboratories are not submitted 8 points are lost.

**The deadline for the first laboratory is Oct 23rd 2015 at 7 p.m.**

**Exercise 1**: build the following directory tree:




modify permissions of directories in order that:

1. everybody can read and write in a;
2. nobody can create subdirectories of d;
3. others can not see inside c, but can create brothers of g;
4. others can enter in g, but not in h;
5. group can not see g (visualize what is inside g), but can enter in h;
6. others has no rights on b, while group has all permissions;
7. owner can read i, but has no other rights on i;

Create a file xyz in folder "a" (using touch command) and modify permissions in order to let the owner read and write it and avoid everyone else. What happens if another user tries to read the file? What happens if another user tries to delete the file?

**Exercise 2** – Umask command

Verify which permissions settings are applied to files and directories when commonly created. Then, using umask command, try to have these files created readable and writable from everyone. Try to have these files readable and writable only from the user. What happens with directories?

**Exercise 3** – Finding files

- Find all files in the system named "core"; try to avoid visualizing errors on the screen (put errors in a log file);
- Find all the files whose name contains the string "conf" and show its size;

- Find all files of the bin user, which are in the /usr directory (and all its subdirectories) and whose permissions are: –r-xr-xr-x;
- Find all files of yours that have been seen from less than a week.

**Exercise 4** – Listing files

Create 15 empty files called [xxx1, xxx2, xxx3 , ..... , xxx15]. Show the list of the previously created file names both in ascending and descending order. Save the list in a file called "list.txt".

**Exercise 5** – Find

Find all the directories of your home whose name ends with the string "backup" (eventually create some of them to check it) and move them in a backup directory.

**Exercise 6** – Find

Find all files in the system that are greater than 2Mbyte or whose name begins with "a" and finishes with "o".

**Exercise 7** – Scripts

Write a script able to read two parameters from the command line. The two parameters represent the names of two directories. Copy all the files from the first directory to the second.

**Exercise 8** – Scripts

Verify that the PATH variable contains the /usr/local/bin directory and print a message with the result of the verification.

**Exercise 9** – Scripts

Read from **stdin** a set of strings, stopping when the string is "END". Copy and number each line on **stdout**.

**Exercise 10** – Scripts

Write a script that given two numbers as parameters from the command line, visualizes a rectangle whose dimensions are the two parameters.

Example:

```
$> ./Create_Rectangle.sh 5 10

→ +--------+
  |        |
  |        |
  |        |
  +--------+
```

**Exercise 11** – Scripts

Write a script that given two parameters from the command line, the first as a directory name and the second as the size in bytes of a file, visualizes all ordinary files in the specified directory that can be read (read permission enabled) and with a size greater than the specified one. Please also verify that the given parameters are correct.

**Exercise 12** – Scripts

Write a script that is able to create the following folder tree:

```
destination ----- project_name ---- branches
                                    |
                                    --- tags
                                    |
                                    --- trunk
```

where the destination directory and the project_name are given by parameters. The script has to stop and notify any issue during the creation steps. Try it several times choosing destinations either writable or read-only.